

파이썬 과학계산

개발환경



<고 강 태> / james@thinkbee.kr

목차

1. 파이썬
2. 과학계산 개발환경
3. conda 개발환경
4. Jupyter Notebook

01. 파이썬

1 Python

- Guido van Rossum이 90년대 발명
- "Monty Python" 에서 'Python'
- 처음 부터 오픈소스로 시작
- Guido 구글 입사후 본격적으로 알려짐
- 스크립트 언어지만 그 이상의 무엇이 있다



"Python is an experiment in how much freedom program-mers need. Too much freedom and nobody can read another's code; too little and expressive-ness is endangered."

- Guido van Rossum

■ 파이썬의 핵심 철학은 PEP 20 문서에 잘 정리되어 있다.

○ <https://www.python.org/dev/peps/pep-0020/>

■ "아름다운게 추한 것보다 낫다." (Beautiful is better than ugly)

■ "명시적인 것이 암시적인 것 보다 낫다." (Explicit is better than implicit)

■ "단순함이 복잡함보다 낫다." (Simple is better than complex)

■ "복잡함이 난해한 것보다 낫다." (Complex is better than complicated)

■ "가독성은 중요하다." (Readability counts)

1 파이썬은 어떤 언어인가?

- 대화기능의 인터프리터 언어로 객체 지향 언어
- 동적 데이터 타입 결정 지원
- 플랫폼에 독립적으로 재사용을 통한 개발기간 단축
- 간단하고 쉬운 문법으로 짧은 코드를 지향
- 뛰어난 확장성(C, C++, Java)
- 메모리 자동관리
- 들여쓰기 문법을 통해 소스 코드 관리를 획기적으로 개선
- Pyrex, Psyco, NumPy 등 수치 연산 라이브러리로 과학 기술 컴퓨팅, 공학 분야서 많이 이용

1 파이썬 애플리케이션

■ GUI 프로그래밍

- Tcl/tk를 이용한 UI, wxPython(Window 인터페이스)

■ 웹 프레임워크 및 툴

- Django, OOM 을 위한 Flask

■ 기본 프로그래밍

- socket, SOAP RPC 지원, 인터넷 프로토콜(FTP, SMTP, HTTP)

■ DB 프로그래밍

- SQLite 내장, Oracle, DB2, Sybase, MySQL 등 각종 유명 DB 인터페이스 제공

Apr 2019	Apr 2018	Change	Programming Language	Ratings	Change
1	1		Java	15.035%	-0.74%
2	2		C	14.076%	+0.49%
3	3		C++	8.838%	+1.62%
4	4		Python	8.166%	+2.36%
5	6	▲	Visual Basic .NET	5.795%	+0.85%
6	5	▼	C#	3.515%	-1.75%
7	8	▲	JavaScript	2.507%	-0.99%
8	9	▲	SQL	2.272%	-0.38%
9	7	▼	PHP	2.239%	-1.98%
10	14	▲▲	Assembly language	1.710%	+0.05%
11	18	▲▲	Objective-C	1.505%	+0.25%
12	17	▲▲	MATLAB	1.285%	-0.17%
13	10	▼	Ruby	1.277%	-0.74%
14	16	▲	Perl	1.269%	-0.26%
15	11	▼▼	Delphi/Object Pascal	1.264%	-0.70%
16	12	▼▼	R	1.181%	-0.63%
17	13	▼▼	Visual Basic	1.060%	-0.74%
18	19	▲	Go	1.009%	-0.17%
19	15	▼▼	Swift	0.978%	-0.56%
20	68	▲▲	Groovy	0.932%	+0.82%



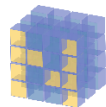
02. 과학계산 개발환경

2 과학계산 개발환경

■ 2



ML Libraries



NumPy



SciPy



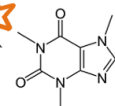
Keras



theano



Spark



TensorFlow

pandas



matplotlib



Environments

Pipenv

Pyenv

virtualenv

venv

CONDA

Distribution

pip



■ 기본 Python interpreter 외에 다양한 언어로 지원

- Cpython: C로 작성된 파이썬 인터프리터
- Jython: Java로 작성된 파이썬 인터프리터
- IronPython: C# 파이썬 인터프리터 (.Net 플랫폼)
- PyPy: 파이썬으로 작성된 파이썬 인터프리터

파이썬과 배포판

파이썬 공식 배포본은 www.python.org 에서 제공하는 설치 바이너리로 CPython 을 기반으로 제작되어 배포된다. 여기에는 인터프리터, 라이브러리 등의 기본 파이썬 유틸리티가 설치된다.

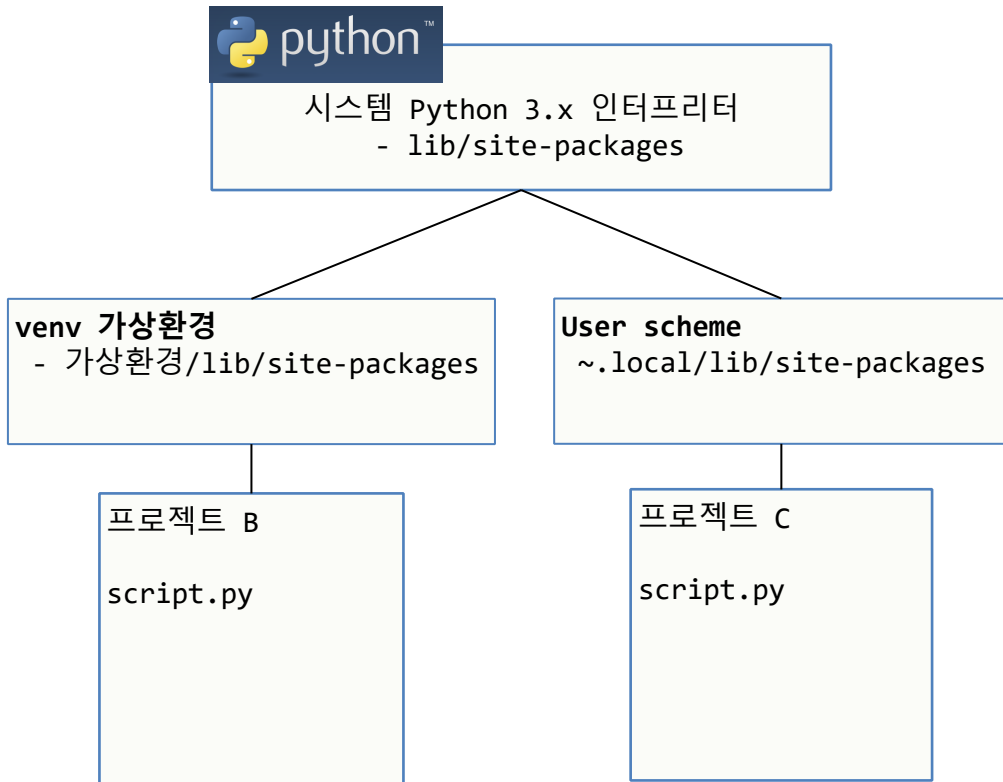
그런데 파이썬은 여러 패키지가 다양한 버전의 파이썬에 연관되어 사용하게 됩니다. 여러 파이썬 버전에 맞게 그때 그때 설치하는게 불편함이 있다. 이를 어려움을 극복하고자 파이썬 실행 환경과 주요 라이브러리를 함께 설치해 사용할 수 있도록 해주는 것이 배포판이다.

■ 일반적인 파이썬 환경 구조



conda 가상환경
- Anaconda/envs/가상환경/
lib/site-packages

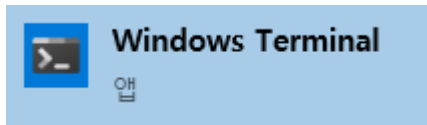
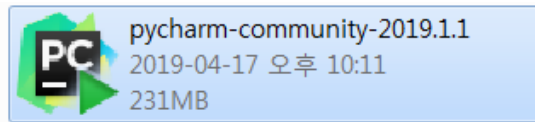
프로젝트 A
- Conda 기반
Anaconda/conda



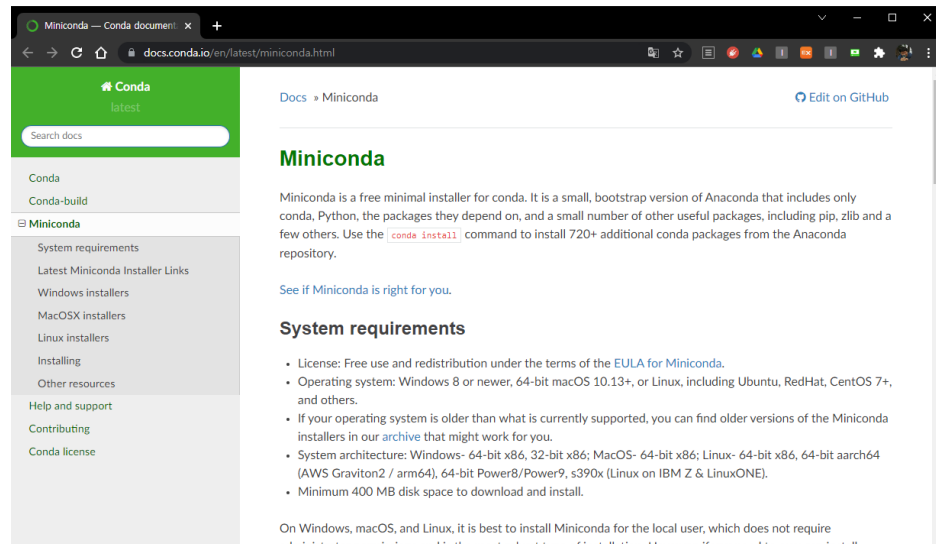
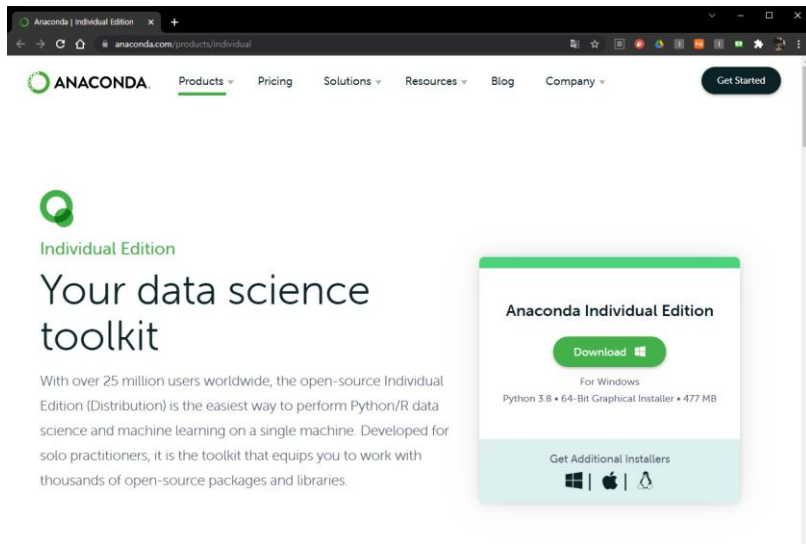
2 과학계산 개발환경

■ 파이썬 코드 작성 도구

- Jetbrain PyCharm community edition
- Visual Studio Code
- Jupyter Notebook
- Spyder



■ Anaconda, Miniconda



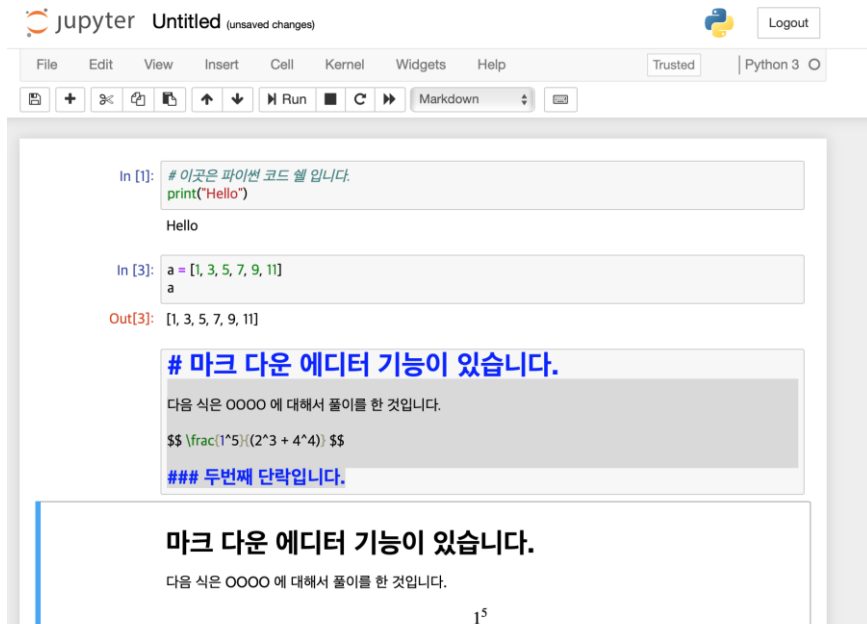
■ 가상환경이란

- 파이썬 사용시 여러 모듈을 설치하는데 보통 시스템에 설치된 파이썬 lib/site-packages 폴더에 위치한다.
 - python 2.x, python 3.x 등 모듈 분리할 필요
 - 과학계산용, 인공지능용 등의 필요로 모듈 분리 필요
- 이런 목적으로 가상개발환경을 이용한다.
 - 파이썬 공식 가상환경 venv
 - 사용자 혹은 벤더가 지원하는 가상환경 모듈
 - ✓Anaconda의 conda

■ 대표적인 가상환경 모듈

- **venv** : Python 3.3 버전 이후 부터 기본모듈에 포함됨
 - <https://docs.python.org/ko/3/tutorial/venv.html>
- **virtualenv** : Python 2 이후, Python 3도 사용가능
- **conda** : Anaconda 배포본 사용시 설치되는 모듈
- **pyenv** : Python Version Manger 기능, 가상환경 기능을 플러그인 형태로 제공

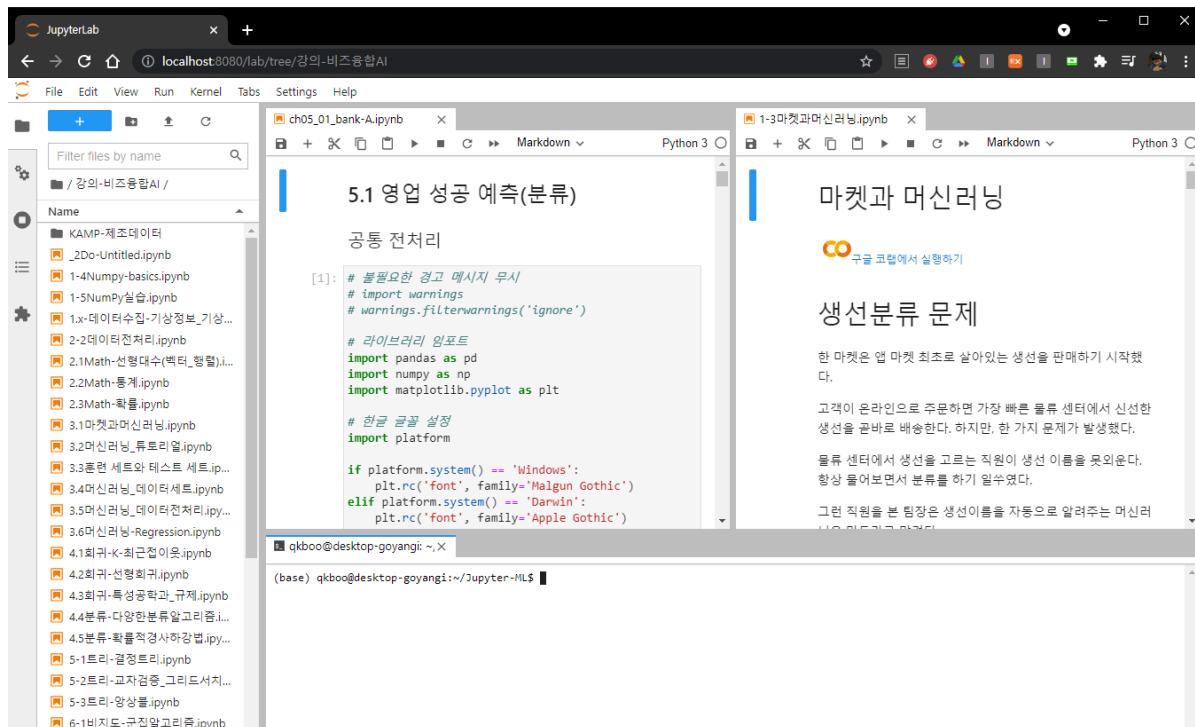
■ Jupyter Notebook 은 웹 기반 GUI 대화형 인터프리터.



```
$ pip install jupyter
```

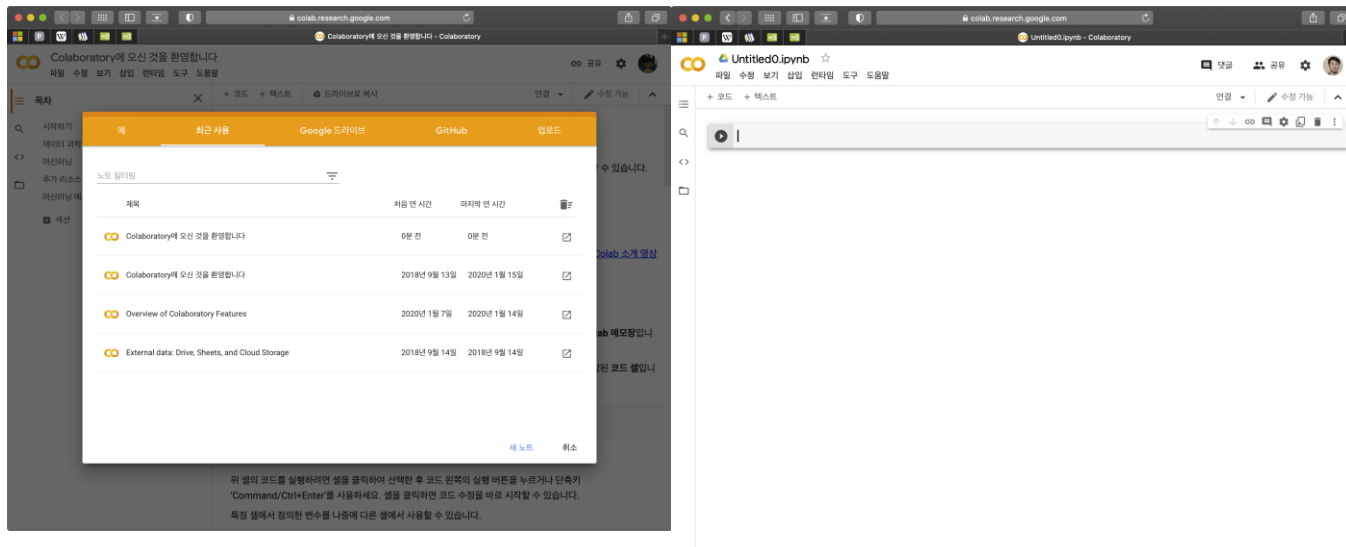
```
$ jupyter notebook --no-browser [DIR]
```

Jupyter Lab



\$ pip install jupyterlab
\$ jupyter-lab --no-browser [DIR]

■ <https://colab.research.google.com/>



■ Windows Terminal

○ Microsoft Store 에서 다운로드

The screenshot shows a Windows Terminal window with a dark background. The title bar indicates the user is 'qkboo@desktop-goyangi' and the active window is 'Windows PowerShell'. The terminal displays the output of a JupyterLab installation process, including messages from the 'ServerApp' and 'LabApp' about validating the language server, starting the JupyterLab extension, and serving the notebook interface. A context menu is open over the terminal text, listing various shells and settings: 'Windows PowerShell' (Ctrl+Shift+1), '명령 프롬프트' (Ctrl+Shift+2), 'Git bash' (Ctrl+Shift+3), 'Azure Cloud Shell' (Ctrl+Shift+4), 'Ubuntu-20.04' (Ctrl+Shift+5), '설정' (Ctrl+,), '피드백', and '정보'.

```
-json-languageserver-bin':
  [<ValidationError: "None is not of type 'string'"]
[W 2021-09-07 01:25:47.355 ServerApp] Failed to validate language-server':
  [<ValidationError: "None is not of type 'string'"]
[I 2021-09-07 01:25:47.396 ServerApp] jupyter_lsp | ext
[I 2021-09-07 01:25:47.397 LabApp] JupyterLab extension
2021.05/lib/python3.8/site-packages/jupyterlab
[I 2021-09-07 01:25:47.397 LabApp] JupyterLab applicat
a3-2021.05/share/jupyter/lab
[I 2021-09-07 01:25:47.399 ServerApp] jupyterlab | ext
[I 2021-09-07 01:25:47.404 ServerApp] nbclassic | ext
[I 2021-09-07 01:25:47.404 ServerApp] Serving notebook
[I 2021-09-07 01:25:47.404 ServerApp] Jupyter Server 1
[I 2021-09-07 01:25:47.404 ServerApp] http://desktop-goyangi:8080/lab?token=45bd9ff7e78477d35f1159dd707a33882dd9e02bd92e8589
[I 2021-09-07 01:25:47.404 ServerApp] or http://127.0.0.1:8080/lab?token=45bd9ff7e78477d35f1159dd707a33882dd9e02bd92e8589
[I 2021-09-07 01:25:47.404 ServerApp] Use Control-C to stop this server and shut down all kernels (twice t
o skip confirmation).
[C 2021-09-07 01:25:47.406 ServerApp]

To access the server, open this file in a browser:
  file:///home/qkboo/.local/share/jupyter/runtime/jpserver-283-open.html
Or copy and paste one of these URLs:
  http://desktop-goyangi:8080/lab?token=45bd9ff7e78477d35f1159dd707a33882dd9e02bd92e8589
  or http://127.0.0.1:8080/lab?token=45bd9ff7e78477d35f1159dd707a33882dd9e02bd92e8589
```

■ Deep Running System

○CPU

- 잘 설계된 파이토치 등 코드는 대부분 GPU 사용량을 최대화
 - ✓ 보통 병렬 연산은 모두 GPU
 - ✓ 일분 순차 연산은 CPU, 1~2개 코어에 집중되 100%
 - ✓ 코어 숫자보다 클럭 높은 것
- 전처리/이베딩 단계는 CPU 코어가 많은게 좋다

○RAM

- 16GB 이상, 보통 64GB

○GPU

- 대부분 딥러닝 프레임워크는 보통 NVidia CUDA 코어 사용
 - ✓ CUDA core 수
 - ✓ Memory 크기: 메모리 클 록 큰 배치 실행.
 - ✓ Memory bandwidth

03. conda 개발환경

- Anaconda
- Miniconda

■ Anaconda의 Python 배포본은 과학기실 모듈을 종합한 프레임워크.

- <https://www.anaconda.com/>

■ anandona에는 가상환경 관리자: conda

- 패키지 관리자 / 주요 지원 패키지
 - ✓ NumPy, SciPy, Matplotlib, pandas
 - ✓ IPython, Jupyter notebook
 - ✓ scikit-learn
 - ✓ Intel MKL library
 - ✓ Enthought Canopy
 - ✓ Python

■ Miniconda는 Anaconda 부트스트랩 버전 conda, Python 관련 패키지만 제공

- conda 및 pip, python basic packages
- 기타 conda 명령으로 Anaconda 저장소에서 패키지를 설치 가능
 - 대략 720개 이상의 과학계산용 패키지
- <https://docs.conda.io/en/latest/miniconda.html>

■ 패키지 관리자: conda

- Python
- conda

W

Python version	Name	Size	
Python 3.9	Miniconda3 Windows 64-bit	58.1 MiB	
Python 3.8	Miniconda3 Windows 64-bit	57.3 MiB	
Python 3.7	Miniconda3 Windows 64-bit	55.8 MiB	
Python 3.9	Miniconda3 Windows 32-bit	55.3 MiB	
Python 3.8	Miniconda3 Windows 32-bit	54.5 MiB	
Python 3.7	Miniconda3 Windows 32-bit	55.3 MiB	

■ Miniconda는 Anaconda 부트스트랩 버전 conda, Python 관련 패키지만 제공

- conda 및 pip, python basic packages
- 기타 conda 명령으로 Anaconda 저장소에서 패키지를 설치 가능
 - 대략 720개 이상의 과학계산용 패키지
- <https://docs.conda.io/en/latest/miniconda.html>

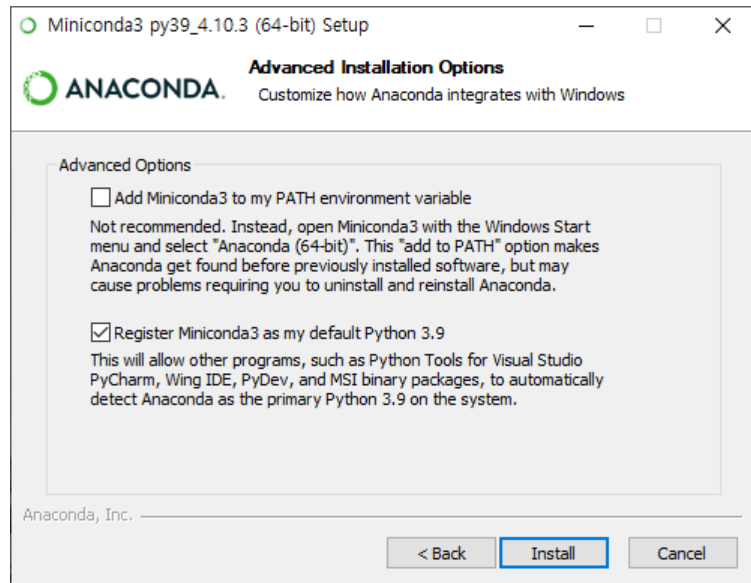
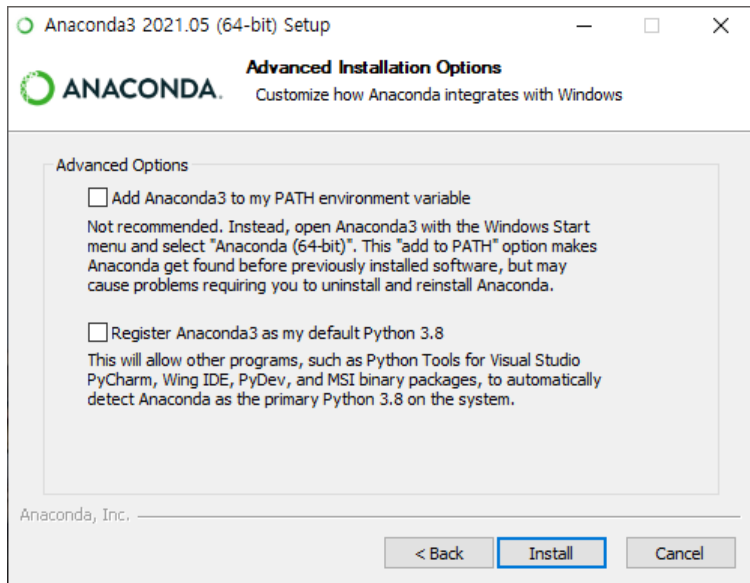
■ 패키지 관리자: conda

- Python
- conda

W

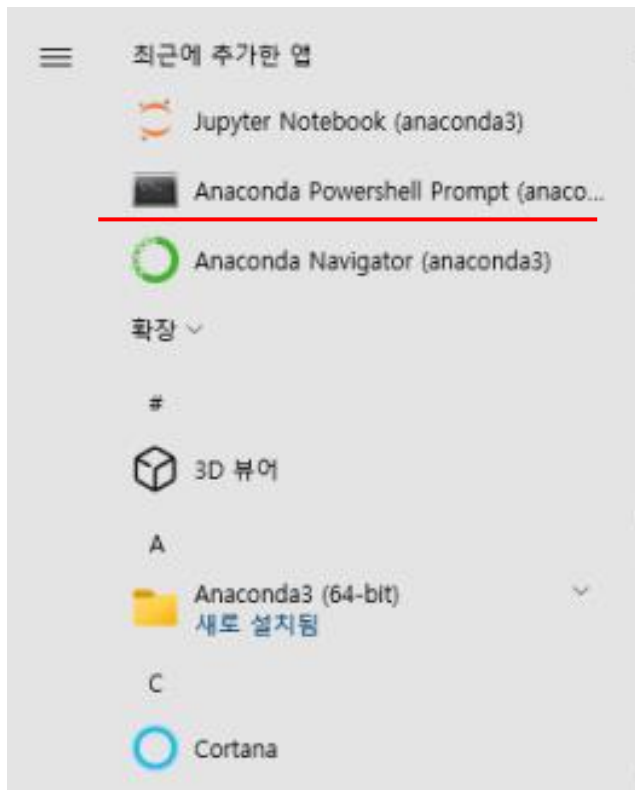
Python version	Name	Size	
Python 3.9	Miniconda3 Windows 64-bit	58.1 MiB	
Python 3.8	Miniconda3 Windows 64-bit	57.3 MiB	
Python 3.7	Miniconda3 Windows 64-bit	55.8 MiB	
Python 3.9	Miniconda3 Windows 32-bit	55.3 MiB	
Python 3.8	Miniconda3 Windows 32-bit	54.5 MiB	
Python 3.7	Miniconda3 Windows 32-bit	55.3 MiB	

- Anaconda 혹은 Miniconda 를 기본설치 진행하면 된다.
 - 단, 경로 추가는 선택하지 말자!



3 Anaconda 개발환경

■ Anaconda 시작



■ conda 명령

conda info: 현재 환경 정보

conda search 패키지 : 패키지 검색

conda install 패키지[=버전] : 패키지 (버전 지정) 설치

conda install 패키지=버전=파이썬 : 파이썬 버전 지정해 패키지 설치

conda --version : 버전 확인

conda update conda : 업데이트

conda update 패키지: 패키지 업데이트

conda list : 설치된 패키지 출력

conda remove 패키지: 패키지 삭제

conda remove --force 패키지: 패키지 삭제

conda list -export > packages.txt 패키지 목록 저장

conda install -file packages.txt 패키지 목록에서 설치

■ 아나콘다 가상 환경 만들기

- 아나콘다에서는 conda를 사용하여 가상 환경을 만듦

이름 지정

```
conda create --name <NAME> python=<version> [anaconda]
```

```
conda create -n <NAME> python=<version>
```

환경 파일

```
conda env create -f environment.yml
```

#특정 위치

```
conda create --prefix 위치 <패키지>
```

- 특정 파이썬 버전은 python=3.5처럼 버전을 지정해줌
- 아나콘다에서 venv를 사용가능, 하지만 conda 가상 환경 사용하는 것을 권장함

- 모듈 업데이트

```
conda update <package-name>
```

- 가상환경 복사하기

```
conda create -n TARGET -clone SRC
```

- 가상환경 삭제하기

```
conda remove -n 가상환경이름 --all
```

■ PIP : Python Index Package

- Python은 pip 로 패키지/모듈을 설치할 수 있다.
- Python은 파이썬 표준 라이브러리(Python Standard Library, PSL) 이외, PyPI(Python Package Index)를 통해 다양한 패키지 설치 가능.

```
pip install <packages/modules name>
pip uninstall <packages/modules name>
pip list
pip show <package_name>
```

<https://www.anaconda.com/blog/using-pip-in-a-conda-environment>

■ PIP 로 설치한 모듈은 다음 위치에 설치 된다:

Platform	Standard installation location	Default value
Unix (pure)	<code>prefix/lib/pythonX.Y/site-packages</code>	<code>/usr/local/lib/pythonX.Y/site-packages</code>
Unix (non-pure)	<code>exec-prefix/lib/pythonX.Y/site-packages</code>	<code>/usr/local/lib/pythonX.Y/site-packages</code>
Windows	<code>prefix\Lib\site-packages</code>	<code>C:\PythonXY\Lib\site-packages</code>

■ - 참조: <https://bit.ly/36N1B3r>

■ PIP 설치 패키지 관리

- 처음 설치후 가상환경 등에서 모듈 혹은 패키지를 설치후 전체 패키지를 추출해 관리
- pip freeze 시스템 패키지 제외하고 설치한 패키지 목록과 버전 정보를 requirements.txt 파일에 저장

```
(firstenv) $ pip freeze > requirements.txt
```

```
(firstenv) $ more requirements.txt  
numpy==1.16.4
```

- 다른 가상환경에서 requirements.txt 패키지 설치

```
(secondenv) $ pip install -r requirements.txt
```

```
(secondenv) $ pip uninstall -r requirements.txt
```

■ 가상환경 생성

```
$ conda create -n DS_p38 python=3.8
```

```
$ conda activate DS_p38
```

```
(DS_p38) $ conda env list
          * DS_p38
          base
```

```
(DS_p38) $ conda install jupyterlab pandas
```

```
...
```

```
(DS_p38) $ conda list
```

```
(DS_p38) $ conda deactivate
```

```
...
```

```
$
```

■ 예) Python 버전 지정 가상환경 생성

```
$ conda create --name simple python=3.6  
$ conda create -n simple python=3.6
```

```
$ conda activate simple  
(simple) $ python --version  
Python 3.6
```

```
$ conda info --envs  
# conda environments:  
base * /anaconda3  
simple /anaconda3/envs/simple
```

■ 예) base 환경 기반 가상환경 생성

```
$ conda create -n ml python=3.8 --clone base
```

```
$ conda activate ml
```

```
(ml) $ python --version
```

```
Python 3.8
```

```
$ conda info --envs
```

```
# conda environments:
```

```
base * /anaconda3
```

```
simple /anaconda3/envs/simple
```

```
ml /anaconda3/envs/ml
```

3 Anaconda 개발환경

■ 아나콘다 가상 환경 위치

- conda는 아나콘다 설치 폴더의 envs 안에 생성함

```
C:\Users\qkboo\Anaconda3\envs\tf2
```

- CMD/PS 에서 직접 실행시 가상환경의 Scripts\activate 에 가상 환경 이름을 지정

```
C:\Users\qkboo\Anaconda3\Scripts\activate tf2  
(tf2) C:\
```

■ pip와 conda

- 아나콘다 가상 환경에 패키지를 설치할 때는 pip 대신 conda를 사용해야 함
 - pip를 사용하면 아나콘다 설치 폴더의 Anaconda\Lib\site-packages 안에 패키지가 저장되므로 주의해야 함

4. Jupyter Notebook

- 01. Notebook 실행
- 02. 기본 기능 이해
- 03. Ipython 모듈
- 04. Magic Command
- 05. LaTeX 사용하기

01. Notebook 실행

■ Jupyter Notebook 과 Jupyterlab

- Anaconda 는 jupyter 와 jupyterlab 을 모두 제공

```
(base) PS C:\Users\Wandro> conda list jupyter
# packages in environment at C:\Users\Wandro\anaconda3:
#
# Name                        Version                        Build      Channel
jupyter                       1.0.0                        py38_7
jupyter-packaging            0.7.12                       pyhd3eb1b0_0
jupyter_client               6.1.12                       pyhd3eb1b0_0
jupyter_console              6.4.0                        pyhd3eb1b0_0
jupyter_core                  4.7.1                        py38haa95532_0
jupyter_server                1.4.1                        py38haa95532_0
jupyterlab                    3.0.14                       pyhd3eb1b0_1
jupyterlab_pygments          0.1.2                        py_0
jupyterlab_server            2.4.0                        pyhd3eb1b0_0
jupyterlab_widgets           1.0.0                        pyhd3eb1b0_1
(base) PS C:\Users\Wandro>
```


1 Jupyter notebook 커널 실행

■ Jupyter Notebook 과 Jupyterlab

```
$ jupyter notebook --no-browser --ip=* DIRECTORY
```

```
$ jupyter-lab --no-browser --ip=* DIRECTORY
```

The screenshot shows the JupyterLab web interface. At the top is a browser address bar with the URL `111.234.555.222:8000/tree`. Below the browser bar is the JupyterLab header with the logo, a 'Quit' button, and a 'Logout' button. The main interface has three tabs: 'Files' (selected), 'Running', and 'Clusters'. Below the tabs is a message: 'Select items to perform actions on them.' To the right of this message are buttons for 'Upload', 'New', and a refresh icon. The main content area displays a file browser view for the root directory '/'. It shows a list of folders with checkboxes on the left, folder names in the center, and last modified times on the right. The folders are: '_Lectures' (a month ago), '_Notebook' (12 days ago), 'Algorighm' (2 months ago), 'Crawling' (25 days ago), 'data_analysis-master' (2 months ago), 'Data_Analysis_with_Pandas_Python' (2 months ago), 'DataAnalysis_python' (3 months ago), 'DataAnalysis_with_Python' (a year ago), 'Datascience_notebook' (2 days ago), 'dataset' (9 days ago), 'Libraries' (a month ago), 'Math_study' (18 days ago), and 'ML Math' (9 days ago).

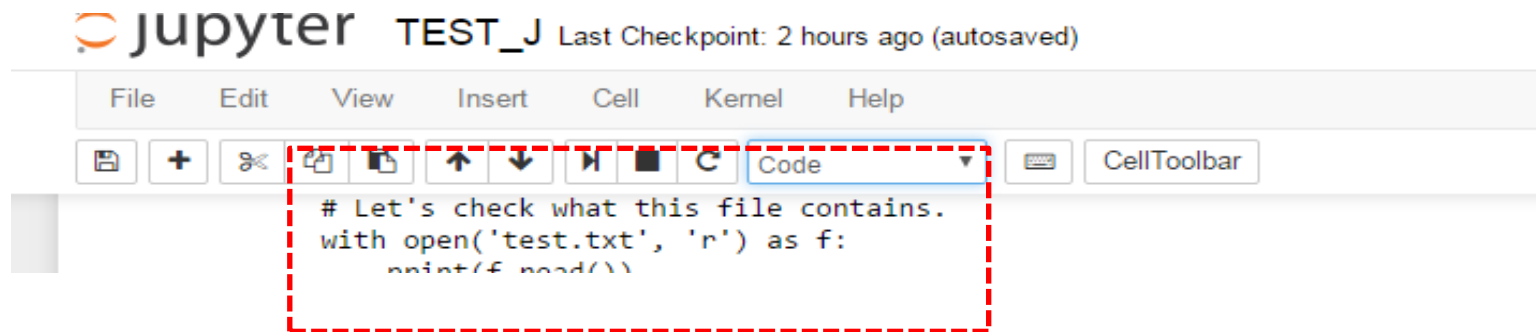
	Name	Last Modified	File size
<input type="checkbox"/>	/		
<input type="checkbox"/>	_Lectures	a month ago	
<input type="checkbox"/>	_Notebook	12 days ago	
<input type="checkbox"/>	Algorighm	2 months ago	
<input type="checkbox"/>	Crawling	25 days ago	
<input type="checkbox"/>	data_analysis-master	2 months ago	
<input type="checkbox"/>	Data_Analysis_with_Pandas_Python	2 months ago	
<input type="checkbox"/>	DataAnalysis_python	3 months ago	
<input type="checkbox"/>	DataAnalysis_with_Python	a year ago	
<input type="checkbox"/>	Datascience_notebook	2 days ago	
<input type="checkbox"/>	dataset	9 days ago	
<input type="checkbox"/>	Libraries	a month ago	
<input type="checkbox"/>	Math_study	18 days ago	
<input type="checkbox"/>	ML Math	9 days ago	

■ New Notebook

The screenshot displays the JupyterLab web interface. At the top, there are 'Quit' and 'Logout' buttons. Below them are tabs for 'Files', 'Running', and 'Clusters'. A message states 'Select items to perform actions on them.' Below this is a file browser showing a directory structure with a folder named '_Notebook' and several files including 'Anaconda설치.ipynb', 'JupyterLab-LSP.ipynb', 'JupyterLab-Plotly.ipynb', and 'JupyterLab-URLs.ipynb'. A 'New' button is visible, and its dropdown menu is open, showing options: 'Notebook:', 'Python 3', 'Python27', 'Python38', 'Other:', 'Text File', 'Folder', and 'Terminal'. The bottom part of the image shows a browser window with the address bar displaying '220.121.133.223:8185/my/notebooks/_Notebo...'. The JupyterLab interface within the browser shows a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. On the right, there are buttons for 'Trusted' and 'Python 3'. The main area contains a code editor with the prompt 'In []: |'.

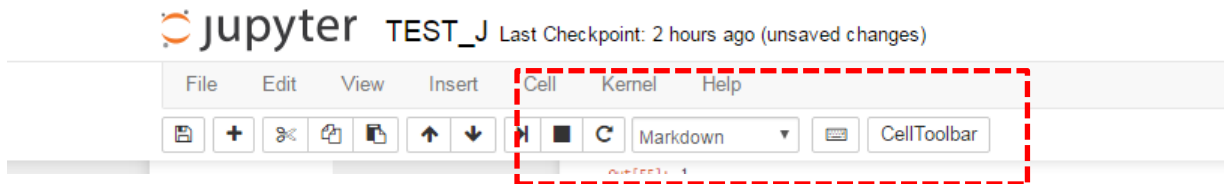
1 Cell type : code

Cell에 Python 코드가 입력되어 실행



1 Cell type : markdown

Cell에 markdown에 대한 표기법으로 수식식이나 문서 등을 작성



! 또는 %(매직 commad)를 이용해서 notebook 파일 보기

```
In [39]: % cd ..
```

```
C:\Users\06411
```

```
In [40]: !ls *.ipynb
```

```
TEST_J.ipynb  
Untitled.ipynb  
dahlmoon.ipynb  
test.ipynb
```

```
In [41]: %ls *.ipynb
```

```
C 드라이브의 볼륨: SYSTEM  
볼륨 일련 번호: 4A5F-4E72
```

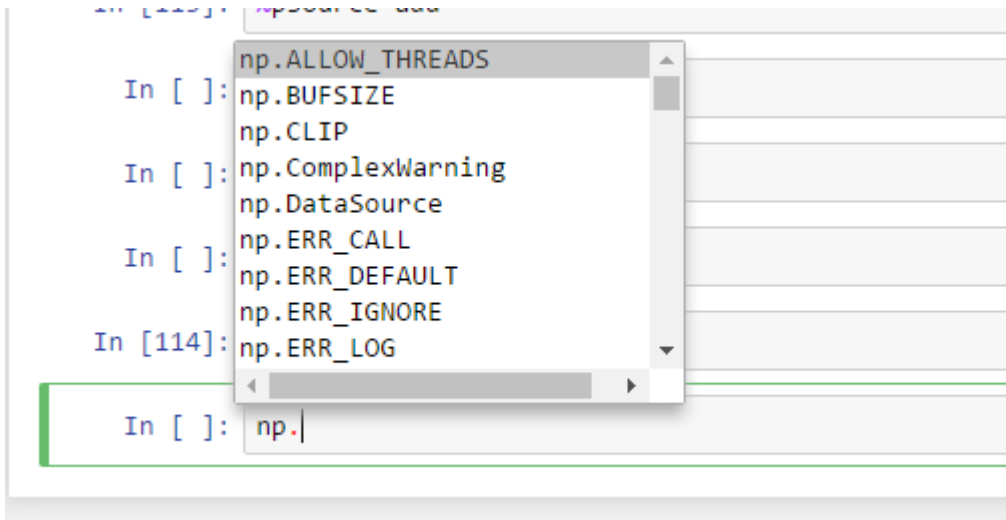
```
C:\Users\06411 디렉터리
```

2016-05-02	오후 01:39	82,213	dahlmoon.ipynb
2016-06-27	오전 11:34	10,624	test.ipynb
2016-06-27	오후 01:16	34,883	TEST_J.ipynb
2016-06-27	오전 11:05	72	Untitled.ipynb
	4개 파일	127,792	바이트
	0개 디렉터리	41,370,497,024	바이트 남음

02. 기본 기능 이해

2 자동완성: tab

ipython 처럼 입력하고 tab 키를 누르면 내부에 있는 요소들을 보여주므로 선택해서 사용 가능

A screenshot of an IPython terminal window demonstrating tab completion. The prompt 'In []: np.' is shown at the bottom. A dropdown menu is open, listing various NumPy attributes: np.ALLOW_THREADS, np.BUFSIZE, np.CLIP, np.ComplexWarning, np.DataSource, np.ERR_CALL, np.ERR_DEFAULT, np.ERR_IGNORE, and np.ERR_LOG. The list is scrollable, with a scrollbar visible on the right. The background shows previous input lines like 'In [114]: np.ERR_LOG' and 'In []: np.'.

```
In [ ]: np.  
In [ ]: np.ALLOW_THREADS  
In [ ]: np.BUFSIZE  
In [ ]: np.CLIP  
In [ ]: np.ComplexWarning  
In [ ]: np.DataSource  
In [ ]: np.ERR_CALL  
In [ ]: np.ERR_DEFAULT  
In [114]: np.ERR_IGNORE  
In [114]: np.ERR_LOG  
In [ ]: np.
```


2 객체 정보 조회 : shift+TAB

변수를 키인하고 shift+TAB을 누르면 내부 특성이 조회 됨

```
import numpy as np
```

```
x = [1, 2, 3, 99, 99, 3, 2, 1]
```

```
x
```

Type: list

String form: [1, 2, 3, 99, 99, 3, 2, 1]

Length: 8

Docstring:

^ + x

2 함수 정보 조회 : shift+TAB

함수를 키인하고 shift+TAB을 누르면 내부 특성이 조회 됨

```
def add(x:int, y:int) ->int :  
    return x+y
```

add

Signature: add(x:int, y:int) -> int

Docstring: <no docstring>

File: c:\users\06411\documents\<ipython-input-820-5bfdc7793909>

Type: function

ipython 처럼 입력한 후 ?를 붙이고 실행시키면 내부 정보가 보임

```
In [115]: list?
```

```
In [ ]: |
```

Docstring:

list() -> new empty list

list(iterable) -> new list initialized from iterable's items

Type: type

2 객체 정보 조회 : ??

정의된 객체에 다음에 ??를 이용하면 help 정보(소스)가 나옴

```
In [117]: add??
```

```
In [ ]:
```

```
Signature: add(x, y)
```

```
Source:
```

```
def add(x,y) :  
    return x+y
```

```
File:      c:\users\06411\test\<ipython-input-111-b603eada704c>
```

```
Type:      function
```

함수를 정의하고 사전에 함수를 매핑해서 실행

```
In [63]: def square(x):  
         """Square of x."""  
         return x*x  
  
         def cube(x):  
             """Cube of x."""  
             return x*x*x
```

```
In [64]: # create a dictionary of functions  
  
funcs = {  
    'square': square,  
    'cube': cube,  
}
```

```
In [65]: x = 2  
  
print square(x)  
print cube(x)  
  
for func in sorted(funcs):  
    print func, funcs[func](x)  
  
4  
8  
cube 8  
square 4
```

2 File 저장 및 처리

텍스트 파일을 생성한 후에 file을 오픈한 후에 처리하기

- %pwd : 현재 디렉토리

```
[7]: %pwd
```

```
[7]: '/home/qkboo/Home/Jupyter-Notebook/Pandas'
```

```
[10]: %cd ../
```

```
/home/qkboo/Home/Jupyter-Notebook
```

```
[11]: %%writefile testfile
```

```
안녕하세요.
```

```
Writing testfile
```

```
[12]: with open('testfile', 'r') as f:
```

```
    print(f.read())
```

```
안녕하세요.
```

%run(매직 commad)을 이용해서 파이썬 모듈 실행

```
In [9]: %%writefile add.py
```

```
def add(x,y) :  
    return x+y  
  
print add(4,4)
```

```
Writing add.py
```

```
In [10]: %ls
```

```
C 드라이브의 볼륨: SYSTEM  
볼륨 일련 번호: 4A5F-4E72
```

```
C:\Users\06411\TEST 디렉터리
```

```
2016-06-27 오후 12:32 <DIR>      .  
2016-06-27 오후 12:32 <DIR>      ..  
2016-06-27 오후 12:32          50 add.py  
                  1개 파일          50 바이트  
                  2개 디렉터리 41,355,968,512 바이트 남음
```

```
In [11]: %run add.py
```

```
8
```

마크다운 셀을 사용해서 로직이나 다양한 설명을 작성함

```
In [ ]: ### New paragraph
This is rich text with [links](http://ipython.org),
equations:

$$\hat{f}(\xi) = \int_{-\infty}^{+\infty} f(x) \mathrm{e}^{-i \xi x} dx$$

code with syntax highlighting:
```python
print("Hello world!")
```

### New paragraph

This is *rich* **text** with [links](http://ipython.org), equations:

$$\hat{f}(\xi) = \int_{-\infty}^{+\infty} f(x) \mathrm{e}^{-i \xi x} dx$$

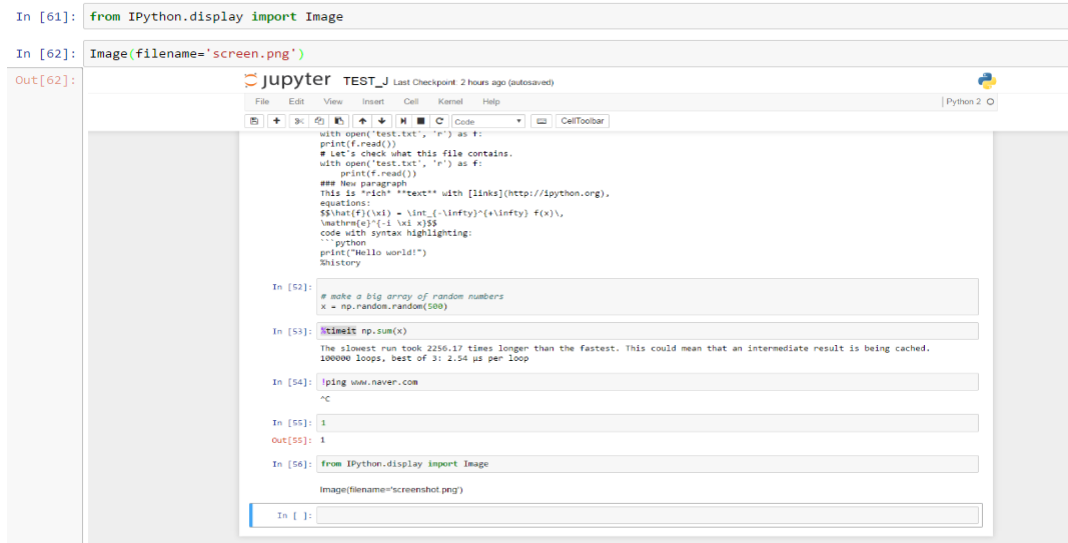
code with syntax highlighting: ```python print("Hello world!")



## 03. Ipython 모듈

---

## Jupyter notebook 이미지를 캡처해서 저장하고 이를 불러 출력

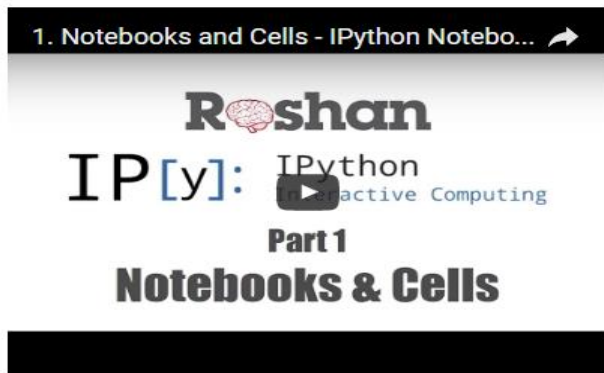


Jupyter notebook에서 유튜브 영상을 처리할 수 있음, 유튜브의 파일명만 내부에 작성하면 호출이 됨

```
In [81]: from IPython.display import YouTubeVideo
```

```
In [82]: YouTubeVideo("lmoNmy-cmSI")
```

```
Out[82]:
```



Jupyter notebook에서 ipython 단축키를 사용  
일부는 Windows에서는 실행되지 않을 수 있음.

Ctrl+P, 위 화살표 키	명령어 이력을 역순으로 검색
Ctrl+N, 아래 화살표 키	명령어 이력을 최근 순으로 검색
Ctrl+R	readline 명령어 형식의 이력 검색
Ctrl+Shift + V	클립보드에서 텍스트 붙여넣기
Ctrl+C	현재 실행중인 코드 중단하기
Ctrl+A	커서를 줄의 처음으로 이동하기
Ctrl+E	커서를 줄의 마지막으로 이동하기
Ctrl+K	커서가 놓인 곳부터 줄이 마지막까지 지우기
Ctrl+U	현재 입력된 모든 텍스트 지우기
Ctrl+F	커서의 앞으로 한글자씩 이동하기
Ctrl+B	커서를 뒤로 한글자씩 이동하기
Ctrl+L	화면 지우기

cell에 표현식을 입력하고 ctrl+ enter를치면 현재 cell이 실행되고 다음 cell이 활성화 되지 않음

```
In [160]: c = 10
```

```
In [161]: print c
```

```
10
```

cell에 표현식을 입력하고 shift+ enter(Alt-Enter )를치면 다음 셀이 활성화

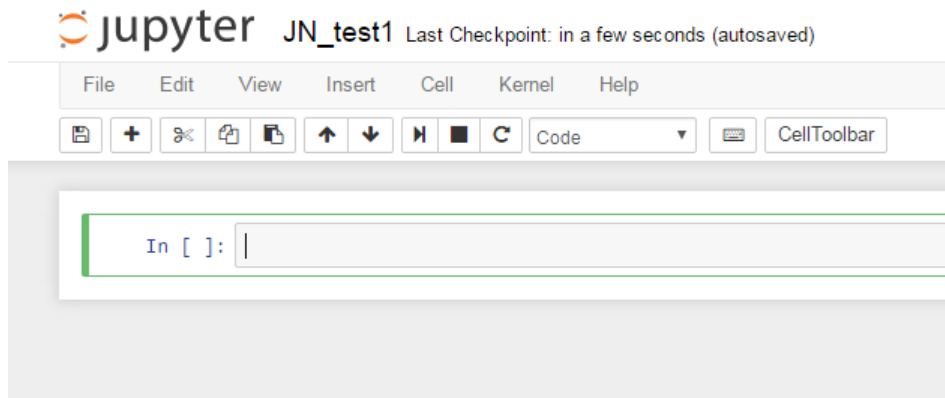
```
In [155]: a=10
```

```
In [156]: print a
```

```
10
```

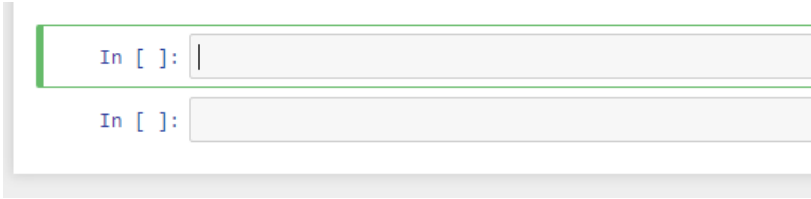
```
In []:
```

현재 jupyter notebook에 저장되었다는 표시가 남음



### 3 단축키: ctrl + shift + "-"

편집모드에서 cell 분할

A screenshot of a Jupyter Notebook interface. It shows two input cells, each starting with the prompt 'In [ ]:'. The top cell has a vertical cursor at the end of the line. The bottom cell is empty. The interface has a light gray background with a green border around the input area.

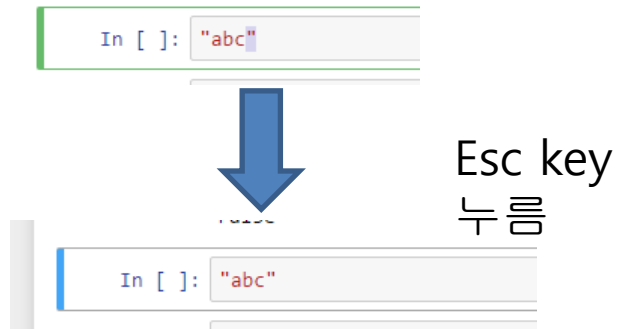
```
In []: |
```

```
In []:
```



### 3 단축키:Esc

편집모드에서 cell을 명령모드로 변경  
초록색에서 파란색으로 변경



## 디렉토리 내부에 특정 파일을 찾아서 표시하기

```
In [188]: files = !ls -l -S | grep .edges
```

```
In [189]: files
```

```
Out[189]: ['-rw-r--r-- 1 06411 Administ 600274 Dec 29 2012 1912.edges',
 ['-rw-r--r-- 1 06411 Administ 523802 Dec 29 2012 107.edges',
 ['-rw-r--r-- 1 06411 Administ 280354 Dec 29 2012 1684.edges',
 ['-rw-r--r-- 1 06411 Administ 96188 Dec 29 2012 3437.edges',
 ['-rw-r--r-- 1 06411 Administ 51066 Dec 29 2012 348.edges',
 ['-rw-r--r-- 1 06411 Administ 37228 Dec 29 2012 0.edges',
 ['-rw-r--r-- 1 06411 Administ 27082 Dec 29 2012 414.edges',
 ['-rw-r--r-- 1 06411 Administ 26496 Dec 29 2012 686.edges',
 ['-rw-r--r-- 1 06411 Administ 4320 Dec 29 2012 698.edges',
 ['-rw-r--r-- 1 06411 Administ 2914 Dec 29 2012 3980.edges']]
```

### 3 Tab 사용해서 이름 검색

파일명 등을 세부 적으로 모를 경우 tab을 이용해서 조회

```
In [194]: !head -n5 1912.bit_length
 1912.circles
 953 1323 1912.conjugate
 1789 1707 1912.denominator
 1175 1059 1912.edges
 1329 1559 1912.imag
 1804 1898 1912.numerator
 1912.real

In []: !head -n5 1912.
```

## 04. Magic command

---

## 4 Magic command

magic command에는 `line(%)`과 `cell(%%)`로 지정해서 처리할 수 있음

line

%magic command

예) %run foo.py 는 s file foo.py를 실행

Cell(전체)

%%magic command

예) %%latex 는 모든 cell에 있는 latex를 번역

## %lsmagic을 이용해서 가지고 있는 command를 전체 조회

```
In [42]: %lsmagic
```

```
Out[42]: Available line magics:
```

```
%alias %alias_magic %autocall %automagic %autosave %bookmark %cd %clear %cls %colors %config %connect_info %copy %ddir %debug %dhist %dirs %doctest_mode %echo %ed %edit %env %gui %hist %history %install_default_config %install_ext %install_profiles %killbgscripts %ldir %less %load %load_ext %loadpy %logoff %logon %logstart %logstate %logstop %ls %lsmagic %macro %magic %matplotlib %mkdir %more %notebook %page %pastebin %pdb %pdef %pdoc %pfile %pinfo %pinfo2 %popd %pprint %precision %profile %prun %psearch %psource %pushd %pwd %pycat %pylab %qtconsole %quickref %recall %rehashx %reload_ext %ren %rep %rerun %reset %reset_selective %rmdir %run %save %sc %set_env %store %sx %system %tb %time %timeit %unalias %unload_ext %who %who_ls %whos %xdel %xmode
```

```
Available cell magics:
```

```
%%! %%HTML %%SVG %%bash %%capture %%cmd %%debug %%file %%html %%javascript %%latex %%perl %%prun %%pypy %%python %%python2 %%python3 %%ruby %%script %%sh %%svg %%sx %%system %%time %%timeit %%writefile
```

```
Automagic is ON, % prefix IS NOT needed for line magics.
```

## 4 주요 Magic command 1

### %lsmagic 내의 주요 명령어 설명

명령어	설명
%pwd, %cd	현재 위치 및 다른 디렉토리로 이동
%history	명령어 히스토리 출력
%reset	모든 정의된 변수 삭제
%%capture	실행되는 명령에 대한 정보의 결과를 저장
%whos	현재 정의된 변수 표시
%pdoc, %psource	Help 기능 실행
%timeit	평균 실행 시간을 출력
%bookmark	디렉토리에 대한 별칭을 저장하고 쉽게 이동할 수 있게 해줌
%%writefile	현재 디렉토리에 파일 생성
%load	디렉토리에 있는 파일을 셀에 로딩
%run	py 프로그램 파일을 실행
%matplotlib inline	matplotlib을 내부 셀에서 실행하기

## %lsmagic 내의 주요 명령어 설명

명령어	설명
%ls	현재 디렉토리에 파일들을 보기
%magic	모든 매직 함수에 대한 상세 도움말 출력
%pdb	예외가 발생하면 자동적으로 디버거 진입.(한번 입력시 ON, 다시 입력시 OFF)
%debug	작성된 코드에 대한 debug 처리



%magic comand 뒤에 ?를 입력하면 설명이 나옴

```
In [45]: %edit?
```

```
In []: |
```

**Docstring:**

Bring up an editor and execute the resulting code.

**Usage:**

%edit [options] [args]

%edit runs an external text editor. You will need to set the command for this editor via the ``TerminalInteractiveShell.editor`` option in your configuration file before it will work.

This command allows you to conveniently edit multi-line code right in your IPython session.

If called without arguments, %edit opens up an empty editor with a temporary file and will execute the contents of this file when you close it (don't forget to save it!).

## magic command에 대한 설명 보기

```
In [118]: %magic
```

```
IPython's 'magic' functions
```

```
=====
```

The magic function system provides a series of functions which allow you to control the behavior of IPython itself, plus a lot of system-type features. There are two kinds of magics, line-oriented and cell-oriented.

Line magics are prefixed with the % character and work much like OS command-line calls: they get as an argument the rest of the line, where arguments are passed without parentheses or quotes. For example, this will time the given statement::

특정 디렉토리를 특정 이름으로 관리하고 싶을때 지정해서 사용

```
%cd facebook
```

```
/notebooks/facebook
```

```
%ls
```

```
0.circles 1684.circles 3437.circles 3980.circles 686.circles
0.edges 1684.edges 3437.edges 3980.edges 686.edges
107.circles 1912.circles 348.circles 414.circles 698.circles
107.edges 1912.edges 348.edges 414.edges 698.edges
```

```
%bookmark fbdata
```

```
%pwd
```

```
u'/notebooks/facebook'
```

```
%cd ..
```

```
/notebooks
```

```
%cd fbdata
```

```
(bookmark:fbdata) -> /notebooks/facebook
/notebooks/facebook
```

%%capture 파일명 후 실제 실행하는 매직명령어의 실행결과를 별도로 저장해서 처리

```
%%capture output
%ls
```

output

```
<IPython.utils.capture.CapturedIO at 0x7fef777fef50>
```

```
output.stdout
```

```
u' withfile.txt \x1b[0m\x1b[01;34mfacebook\x1b[0m/\r\n1_hello_tensorflow.ipynb fo
o.txt\r\n2_getting_started.ipynb fo01.txt\r\n3_mnist_from_scratch.ipynb image.jpg\r\nC:\Users
411\\Downloads\\line_plot_plus.pdf line_plot_plus.pdf\r\nLICENSE mod.py\r\nMatplotlib test1.
ipynb mod.pyc\r\nMover_ch2.pyde mod.fy\r\nUntitled.ipynb
 mod_f.pyc\r\nUntitled1.ipynb newfile.txt\r\nUntitled2.ipynb test.txt\r
\r\nUntitled3.ipynb test2.ipynb\r\nnarraystore.nd test_1.ipynb\r\nndata.txt
 understanding Python_20160815.ipynb\r\nndoctest.py withfile.txt\r
\r\nndoctest.pyc yum-2.0.7.tar.gz\r\n'
```

```
output.show()
```

withfile.txt	facebook/
1_hello_tensorflow.ipynb	foo.txt
2_getting_started.ipynb	foo1.txt
3_mnist_from_scratch.ipynb	image.jpg
C:\Users\411\Downloads\line_plot_plus.pdf	line_plot_plus.pdf
LICENSE	mod.py
Matplotlib_test1.ipynb	mod.pyc
Mover_ch2.pyde	mod_f.py
Untitled.ipynb	mod_f.pyc
Untitled1.ipynb	newfile.txt
Untitled2.ipynb	test.txt
Untitled3.ipynb	test2.ipynb
arraystore.nd	test_1.ipynb
data.txt	understanding_Python_20160815.ipynb
doctest.py	withfile.txt

`%reset`(매직 commad)은 현재 실행되는 notebook 내의 모든 변수를 삭제함

```
In [77]: x
```

```
Out[77]: 10
```

```
In [78]: %reset
```

Once deleted, variables cannot be recovered. Proceed (y/[n])? y

```
In [79]: x
```

```

NameError Traceback (most recent call last)
<ipython-input-79-401b30e3b8b5> in <module>()
----> 1 x

NameError: name 'x' is not defined
```

현재 실행환경 내의 Variables을 표시, similar to Matlab's whos

```
In [100]: %whos
```

Variable	Type	Data/Info
-----		
YouTubeVideo	type	<class 'IPython.lib.display.YouTubeVideo'>
add	function	<function add at 0x0643EF30>

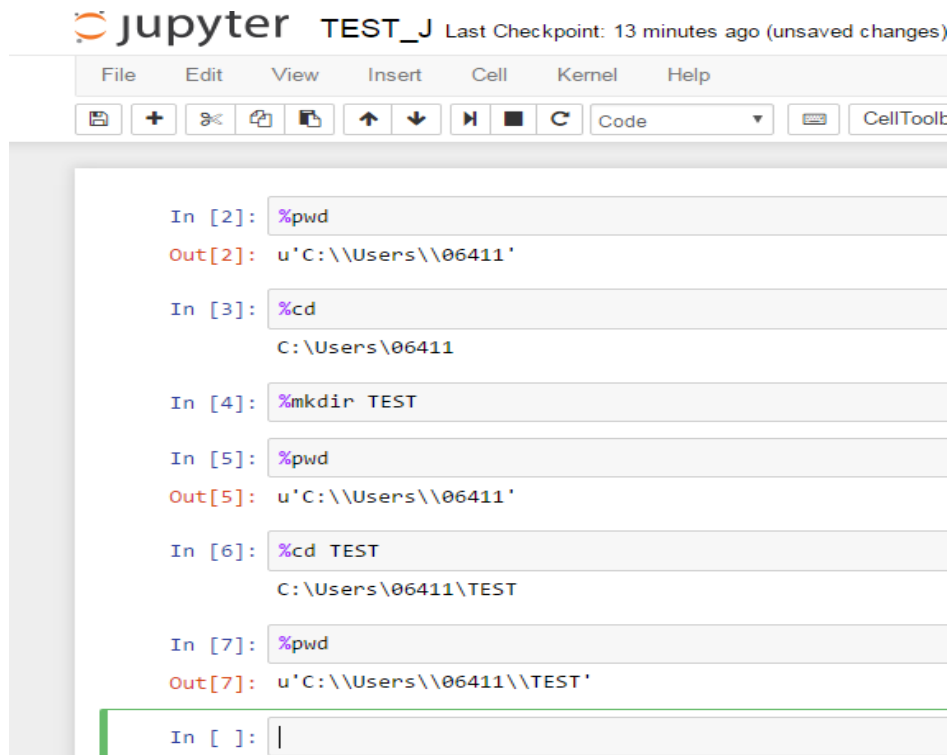
```
In [101]: %reset
```

Once deleted, variables cannot be recovered. Proceed (y/[n])? y

```
In [102]: %whos
```

Interactive namespace is empty.

%(매직 commad)를 이용해서 현재 위치 및 디렉토리 생성 및 이동



The image shows a Jupyter Notebook interface with the title 'TEST\_J' and a status bar indicating 'Last Checkpoint: 13 minutes ago (unsaved changes)'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for file operations and execution. The notebook contains seven input cells with the following commands and their outputs:

```
In [2]: %pwd
Out[2]: u'C:\\Users\\06411'
```

```
In [3]: %cd
 C:\\Users\\06411
```

```
In [4]: %mkdir TEST
```

```
In [5]: %pwd
Out[5]: u'C:\\Users\\06411'
```

```
In [6]: %cd TEST
 C:\\Users\\06411\\TEST
```

```
In [7]: %pwd
Out[7]: u'C:\\Users\\06411\\TEST'
```

```
In []: |
```

%%writefile(매직 commad)를 이용해서 현재 위치에 add.py 생성하고 조회

```
In [9]: %%writefile add.py
```

```
def add(x,y) :
 return x+y

print add(4,4)
```

Writing add.py

```
In [10]: %ls
```

C 드라이브의 볼륨: SYSTEM  
볼륨 일련 번호: 4A5F-4E72

C:\Users\06411\TEST 디렉터리

2016-06-27	오후 12:32	<DIR>	.
2016-06-27	오후 12:32	<DIR>	..
2016-06-27	오후 12:32		50 add.py
		1개 파일	50 바이트
		2개 디렉터리	41,355,968,512 바이트 남음



### %run(매직 commad)을 이용해서 파이썬 모듈 실행

```
In [9]: %%writefile add.py
```

```
def add(x,y) :
 return x+y
```

```
print add(4,4)
```

Writing add.py

```
In [10]: %ls
```

C 드라이브의 볼륨: SYSTEM  
볼륨 일련 번호: 4A5F-4E72

C:\Users\06411\TEST 디렉터리

2016-06-27	오후 12:32	<DIR>	.
2016-06-27	오후 12:32	<DIR>	..
2016-06-27	오후 12:32		50 add.py
		1개 파일	50 바이트
		2개 디렉터리	41,355,968,512 바이트 남음

```
In [11]: %run add.py
```

8

%loadpy(매직 commad)를 이용해서 python 파일을 로드하고 실행하면 결과가  
나옴

```
In []: %loadpy add.py
```

```
In []: # %Load add.py
```

```
def add(x,y) :
 return x+y
```

```
print add(4,4)
```

```
In [23]: # %Load add.py
```

```
def add(x,y) :
 return x+y
```

```
print add(4,4)
```

## 4 %timeit 사용

### %timeit 를 사용해서 실행하는 시간을 점검

```
%history
```

```
In [52]:
```

```
make a big array of random numbers
x = np.random.random(500)
```

```
In [53]: %timeit np.sum(x)
```

The slowest run took 2256.17 times longer than the fastest. This could mean that an irreproducible result was found, or that the test is too slow. Try using the --quiet flag to suppress the usual output, which includes the statistics for the fastest run. Run again. The slowest run took 2256.17 times longer than the fastest. This could mean that an irreproducible result was found, or that the test is too slow. Try using the --quiet flag to suppress the usual output, which includes the statistics for the fastest run. Run again.

```
In [53]:
```

매직커맨드에 ? 를 붙이면 매직커맨드에 대한 help 기능이 실행됨

```
In [424]: %history?
```

```
By default, all input history from the current session is displayed.
Ranges of history can be indicated using the syntax:
```

```
``4``
```

```
Line 4, current session
```

```
``4-6``
```

```
Lines 4-6, current session
```

```
``243/1-5``
```

“%pdoc 객체”를 입력해서 docstring을 조회

```
In [103]: %pdoc list
```

```
In []:
```

```
Class docstring:
 list() -> new empty list
 list(iterable) -> new list initialized from iterable's items
Init docstring:
 x.__init__(...) initializes x; see help(type(x)) for signature
```

“%pdef 객체” , “%psource”를 입력해서 함수의 헤드와 소스를 조회

```
In [111]: def add(x,y) :
 return x+y
```

```
In [112]: %pdef add
 add(x, y)
```

```
In [113]: %psource add
```

```
In []: |
```

```
def add(x,y) :
 return x+y
```

## Cell에 입력된 이력을 출력

In [51]:

```
%history
```

```
%pwd
%pwd
%cd
%mkdir TEST
%pwd
%cd TEST
%pwd
%writefile add.py
```

```
def add(x,y) :
 return x+y
```

```
print add(4,4)
%%writefile add.py
```

```
def add(x,y) :
 return x+y
```

```
print add(4,4)
```

## 현재 명령된 이전 명령 5개만 읽어오기

```
In [192]: %history -l 5
```

```
%ls
```

```
files = !ls -l -S | grep .edges
```

```
files
```

```
%history
```

```
%history?
```



## 4 모든 디렉토리 history 확인

%dhist로 현재까지 방문한 모든 디렉토리 이력을 읽어오기

```
In [193]: %dhist
```

```
Directory history (kept in _dh)
0: C:\Users\06411\TEST
1: C:\Users\06411\TEST\facebook
```

## 4 Cell 입력한 로직 오류 점검

Cell에 입력된 로직에 대한 debug.

```
In [*]: def sub(x,y) :
 aaaaa
 return x + y
```

```
In [*]: %debug
```

```
> <ipython-input-120-ee2d5fc03e64>(2)sub()
1 def sub(x,y) :
----> 2 a
 3 return x + y
```

```
ipdb> |
```

**s(tep)** -- Execute the current line, stop at the first possible occasion (either in a function that is called or in the current function).  
**n(ext)** -- Continue execution until the next line in the current function is reached or it returns.  
**unt(il)** -- Continue execution until the line with a number greater than the current one is reached or until the current frame returns.  
**r(eturn)** -- Continue execution until the current function returns.  
**c(ontinue)** -- Continue execution, only stop when a breakpoint is encountered.

Cell에 !shell command 실행하면 작동되고 이를 파이썬 변수에 할당할 수 있음

```
!ls
```

```
BeautifulSoup_test_20170120.ipynb
GitHub
My Music
My Pictures
My Videos
Processing
__pycache__
add_a_workbook.xlsx
arraystore.nd
arraystore.npy
arraystore.txt
bdsc4j.csv.xlsx
bs4_google_20170130.ipynb
country_data.xml
data.bin
. . .
```

```
contents = !ls
print(contents[:2])
```

```
!cd
path = !cd
print(path)
```

```
!echo "printing from the shell"
message = "hello from Python"
!echo {message}
```

```
['BeautifulSoup_test_20170120.ipynb', 'GitHub']
C:\Users\06411\Documents
['C:\\Users\\06411\\Documents']
"printing from the shell"
hello from Python
```

Cell에서 shell과 magic 명령이 같은 부분이 존재 os별로 상이함

shell	magic	설명
!pwd, !cd	%pwd, %cd	현재 위치 및 다른 디렉토리로 이동
!env	%env	컴퓨터 환경정보 보기
!echo	%echo	메시지 출력하기
!cp	%cp	카피하기
!ls	%ls	현재 디렉토리의 리스트
!mkdir	%mkdir	디렉토리 생성
!rmdir	%rmdir	디렉토리 삭제
!mv	%mv	파일 이동
!rm	%rm	파일 삭제

## 05. LaTeX 사용하기

---

markdown으로 지정하고 \$\$와 \$\$(\$와 \$) 사이에 문자를 입력하고 실행하면 문자가 표현

```
In []: $\alpha, \Alpha, \beta, \Beta, \gamma, \Gamma, \pi, \Pi, \phi, \varphi, \mu, \Phi$
```

$\alpha, \Alpha, \beta, \Beta, \gamma, \Gamma, \pi, \Pi, \phi, \varphi, \mu, \Phi$

## 5 산식 작성 : 제곱과 인덱스

산식을 표현할 때, 제곱(^)과 인덱스(\_{ }) 표시:

```
In []: $k_{n+1} = n^2 + k_n^2 - k_{n-1}$
```

$$k_{n+1} = n^2 + k_n^2 - k_{n-1}$$

## 산식을 표현할 때, 문장과 산식을 표현하고 inline으로 표시

```
In [179]: %%latex
The mass-energy equivalence is described by the famous equation

$$E=mc^2$$

discovered in 1905 by Albert Einstein.
In natural units ($c = 1$), the formula expresses the identity

\begin{equation}
E=m
\end{equation}
```

The mass-energy equivalence is described by the famous equation

$$E = mc^2$$

discovered in 1905 by Albert Einstein. In natural units ( $c = 1$ ), the formula expresses the identity

$$E = m$$



markdown으로 지정하고 \$\$와 \$\$(\$와 \$) 사이에 수학산식을 입력하고 실행하면 실제 수학산식이 표현됨

```
In []: $$c = \sqrt{a^2 + b^2}$$
```

$$c = \sqrt{a^2 + b^2}$$

code로 지정하고 IPython.display 모듈을 이용해서 수학산식을 입력하고 실행하면 실제 수학산식이 표현됨

```
In [150]: from IPython.display import display, Math, Latex
display(Math(r'F(k) = \int_{-\infty}^{\infty} f(x) e^{2\pi i k} dx'))
```

$$F(k) = \int_{-\infty}^{\infty} f(x) e^{2\pi i k} dx$$

Latex를 import하고 latex 정의를 하면 실제 산식으로 표현됨

```
In [152]: from IPython.display import Latex
 Latex(r"""
\begin{eqnarray}
\nabla \times \vec{\mathbf{B}} - \frac{1}{c} \frac{\partial \vec{\mathbf{E}}}{\partial t} &= \frac{4\pi}{c} \vec{\mathbf{j}} \\
\nabla \cdot \vec{\mathbf{E}} &= 4\pi \rho \\
\nabla \times \vec{\mathbf{E}} + \frac{1}{c} \frac{\partial \vec{\mathbf{B}}}{\partial t} &= \vec{\mathbf{0}} \\
\nabla \cdot \vec{\mathbf{B}} &= 0
\end{eqnarray}""")
```

```
Out[152]:
```

$$\left. \begin{aligned} \nabla \times \vec{\mathbf{B}} - \frac{1}{c} \frac{\partial \vec{\mathbf{E}}}{\partial t} &= \frac{4\pi}{c} \vec{\mathbf{j}} \\ \nabla \cdot \vec{\mathbf{E}} &= 4\pi \rho \\ \nabla \times \vec{\mathbf{E}} + \frac{1}{c} \frac{\partial \vec{\mathbf{B}}}{\partial t} &= \vec{\mathbf{0}} \\ \nabla \cdot \vec{\mathbf{B}} &= 0 \end{aligned} \right|$$

## 5 %%latex 이용

### 산식들을 표현하는 예시

```
In [205]: %%latex
\[\int\limits_0^1 x^2 + y^2 dx \]
```

$$\int_0^1 x^2 + y^2 dx$$

```
In [207]: %%latex
\[a_{n_i} \]
```

$$a_{n_i}$$

```
In [209]: %%latex
\[\int_{i=1}^n \]
```

$$\int_{i=1}^n$$

```
In [211]: %%latex
\[\sum_{i=1}^{\infty} \]
```

$$\sum_{i=1}^{\infty}$$

## 5 %%latex 이용

### 산식들을 표현하는 예시

```
In [212]: %%latex
 \[\prod_{i=1}^n \]
```

$$\prod_{i=1}^n$$

```
In [213]: %%latex
 \[\cup_{i=1}^n \]
```

$$\cup_{i=1}^n$$

```
In [214]: %%latex
 \[\cap_{i=1}^n \]
```

$$\cap_{i=1}^n$$

## 5 %%latex 이용

### 산식들을 표현하는 예시

```
In [215]: %%latex
 \[\oint_{i=1}^n\]
```

$$\oint_{i=1}^n$$

```
In [216]: %%latex
 \[\coprod_{i=1}^n\]
```

$$\coprod_{i=1}^n$$

## 5 산식 작성 : 여러 라인 처리

여러 라인 처리는 `\begin{name}`, `\end{name}` 안에 여러 라인의 산식을 표현

```
In [176]: %%latex
\begin{equation}
x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{a_4}}}}
\end{equation}
```

$$x = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{a_4}}}}$$

%%latex로 지정하고 latex 정의를 하면 실제 산식으로 표현됨

```
In [163]: %%latex
 \begin{align}
 \end{align}
```

Latex 시작과 끝

```
In [164]: %%latex
 \begin{align}
 \mathbf{\vec A}
 \end{align}
```

$$\vec{A}$$

```
In [165]: %%latex
 \begin{align}
 \mathbf{\vec A} \cdot \mathbf{\vec B}
 \end{align}
```

$$\vec{A} \cdot \vec{B}$$

```
In [167]: %%latex
 \begin{align}
 \mathbf{\vec A} \cdot \mathbf{\vec B} = \sum_{i=1}^n A_i B_i
 \end{align}
```

$$\vec{A} \cdot \vec{B} = \sum_{i=1}^n A_i B_i$$

수학 산식 추가



## 5 산식 처리 예시 : 단일 라인

%%latex로 지정하고 latex 정의(w[ w])를 하면 실제 산식을 한 라인으로 표현됨

```
In [177]: %%latex
```

```
\[x^n + y^n = z^n \]
```

$$x^n + y^n = z^n$$

%%latex로 지정하고 latex 정의(\begin \end)를 하면 실제 여러줄 산식으로 표현됨

```
In [151]: %%latex
\begin{align}
\nabla \times \vec{\mathbf{B}} &= \frac{1}{c} \frac{\partial \vec{\mathbf{E}}}{\partial t} \\
\nabla \cdot \vec{\mathbf{E}} &= 4\pi \rho \\
\nabla \times \vec{\mathbf{E}} &= -\frac{1}{c} \frac{\partial \vec{\mathbf{B}}}{\partial t} \\
\nabla \cdot \vec{\mathbf{B}} &= 0
\end{align}
```

$$\left. \begin{aligned} \nabla \times \vec{\mathbf{B}} &= \frac{1}{c} \frac{\partial \vec{\mathbf{E}}}{\partial t} \\ \nabla \cdot \vec{\mathbf{E}} &= 4\pi \rho \\ \nabla \times \vec{\mathbf{E}} &= -\frac{1}{c} \frac{\partial \vec{\mathbf{B}}}{\partial t} \\ \nabla \cdot \vec{\mathbf{B}} &= 0 \end{aligned} \right|$$

%%latex로 지정하고 latex 정의를 하면 실제 산식으로 표현됨

```
In [170]: %%latex
\begin{align}
\dot{x} &= \sigma(y-x) \\\
\dot{z} &= -\beta z + xy \\\
\end{align}
```

$$\begin{aligned} \dot{x} &= \sigma(y-x) \\ \dot{z} &= -\beta z + xy \end{aligned}$$

%%latex로 지정하고 latex 정의를 하면 실제 산식으로 표현됨

```
In [172]: %%latex
\begin{equation}
\int f(x) \, \text{d}x = F(x)
\label{eq1}
\tag{eq1text}
\end{equation}
```

$$\int f(x) \, dx = F(x)$$

(eq1text)

## 문자를 표시할 때 사용

## Greek letters

$\alpha A$	<code>\alpha A</code>	$\nu N$	<code>\nu N</code>
$\beta B$	<code>\beta B</code>	$\xi \Xi$	<code>\xi \Xi</code>
$\gamma \Gamma$	<code>\gamma \Gamma</code>	$o O$	<code>o O</code>
$\delta \Delta$	<code>\delta \Delta</code>	$\pi \Pi$	<code>\pi \Pi</code>
$\epsilon \epsilon E$	<code>\epsilon \epsilon E</code>	$\rho \varrho P$	<code>\rho \varrho P</code>
$\zeta Z$	<code>\zeta Z</code>	$\sigma \Sigma$	<code>\sigma \Sigma</code>
$\eta H$	<code>\eta H</code>	$\tau T$	<code>\tau T</code>
$\theta \vartheta \Theta$	<code>\theta \vartheta \Theta</code>	$\upsilon \Upsilon$	<code>\upsilon \Upsilon</code>
$\iota I$	<code>\iota I</code>	$\phi \varphi \Phi$	<code>\phi \varphi \Phi</code>
$\kappa K$	<code>\kappa K</code>	$\chi X$	<code>\chi X</code>
$\lambda \Lambda$	<code>\lambda \Lambda</code>	$\psi \Psi$	<code>\psi \Psi</code>
$\mu M$	<code>\mu M</code>	$\omega \Omega$	<code>\omega \Omega</code>

## 기호를 표시할 때 사용

## Miscellaneous symbols

$\infty$	<code>\infty</code>	$\forall$	<code>\forall</code>
$\Re$	<code>\Re</code>	$\Im$	<code>\Im</code>
$\nabla$	<code>\nabla</code>	$\exists$	<code>\exists</code>
$\partial$	<code>\partial</code>	$\nexists$	<code>\nexists</code>
$\emptyset$	<code>\emptyset</code>	$\varnothing$	<code>\varnothing</code>
$\wp$	<code>\wp</code>	$\complement$	<code>\complement</code>
$\neg$	<code>\neg</code>	$\cdots$	<code>\cdots</code>
$\square$	<code>\square</code>	$\surd$	<code>\surd</code>
$\blacksquare$	<code>\blacksquare</code>	$\triangle$	<code>\triangle</code>

## 기호를 표시할 때 사용

## Binary Operation/Relation Symbols

$\times$	<code>\times</code>	$\otimes$	<code>\otimes</code>
$\div$	<code>\div</code>	$\cap$	<code>\cap</code>
$\cup$	<code>\cup</code>	$\neq$	<code>\neq</code>
$\leq$	<code>\leq</code>	$\geq$	<code>\geq</code>
$\in$	<code>\in</code>	$\perp$	<code>\perp</code>
$\notin$	<code>\notin</code>	$\subset$	<code>\subset</code>
$\simeq$	<code>\simeq</code>	$\approx$	<code>\approx</code>
$\wedge$	<code>\wedge</code>	$\vee$	<code>\vee</code>
$\oplus$	<code>\oplus</code>	$\otimes$	<code>\otimes</code>
$\Box$	<code>\Box</code>	$\boxtimes$	<code>\boxtimes</code>
$\equiv$	<code>\equiv</code>	$\cong$	<code>\cong</code>

## 기호를 표시할 때 사용

LaTeX markup	Renders as
<code>\big( \Big( \bigg( \Bigg(</code>	$(((($
<code>\big) \Big) \bigg) \Bigg)</code>	$)]])$
<code>\big\{ \Big\{ \bigg\{ \Bigg\{</code>	$\{\{\{\{$
<code>\big \langle \Big \langle \bigg \langle \Bigg \langle</code>	$\langle\langle\langle\langle$
<code>\big \rangle \Big \rangle \bigg \rangle \Bigg \rangle</code>	$\rangle\rangle\rangle\rangle$



## 기호를 표시할 때 사용

L <sup>A</sup> T <sub>E</sub> X code	Description
<code>\quad</code>	space equal to the current font size (= 18 mu)
<code>\,</code>	3/18 of <code>\quad</code> (= 3 mu)
<code>\:</code>	4/18 of <code>\quad</code> (= 4 mu)
<code>\;</code>	5/18 of <code>\quad</code> (= 5 mu)
<code>\!</code>	-3/18 of <code>\quad</code> (= -3 mu)
<code>\ (space after backslash!)</code>	equivalent of space in normal text
<code>\quad\quad</code>	twice of <code>\quad</code> (= 36 mu)

## 기호를 표시할 때 사용

Alignment	Environment	Switch command	ragged2e environment	ragged2e switch command
Left	flushleft	\raggedright	FlushLeft	\RaggedRight
Right	flushright	\raggedleft	FlushRight	\RaggedLeft
Centre	center	\centering	Center	\Centering
Fully justified			justify	\justify

## 기호를 표시할 때 사용

## Arrows

$\leftarrow$	<code>\leftarrow</code>	$\Leftrightarrow$	<code>\Leftrightarrow</code>
$\rightarrow$	<code>\rightarrow</code>	$\Rightarrow$	<code>\Rightarrow</code>
$\leftrightarrow$	<code>\leftrightarrow</code>	$\rightleftharpoons$	<code>\rightleftharpoons</code>
$\uparrow$	<code>\uparrow</code>	$\downarrow$	<code>\downarrow</code>
$\Uparrow$	<code>\Uparrow</code>	$\Downarrow$	<code>\Downarrow</code>
$\Leftrightarrow$	<code>\Leftrightarrow</code>	$\Updownarrow$	<code>\Updownarrow</code>
$\mapsto$	<code>\mapsto</code>	$\longmapsto$	<code>\longmapsto</code>
$\nearrow$	<code>\nearrow</code>	$\searrow$	<code>\searrow</code>
$\swarrow$	<code>\swarrow</code>	$\nwarrow$	<code>\nwarrow</code>
$\leftharpoonup$	<code>\leftharpoonup</code>	$\rightharpoonup$	<code>\rightharpoonup</code>
$\leftharpoondown$	<code>\leftharpoondown</code>	$\rightharpoondown$	<code>\rightharpoondown</code>
$\rightleftharpoons$	<code>\rightleftharpoons</code>		